Implementasi Algoritma Dynamic Programming untuk Multiple Constraints Knapsack Problem

(Studi Kasus : Pemilihan Media Promosi di UMN)

Maria Irmina Prasetiyowati
Teknik Informatika
Universitas Multimedia Nusantara
Tangerang Selatan, Indonesia
maria@umn.ac.id

Arya Wicaksana
Teknik Informatika
Universitas Multimedia Nusantara
Tangerang Selatan, Indonesia
kuncikehidupan@windowslive.com

***Abstrak*—Penelitian yang dilakukan membahas mengenai implementasi algoritma *Dynamic Programming* untuk permasalahan *Multiple Constraints Knapsack Problem* (MCKP). MCKP merupakan permasalahan optimasi yang kompleks dalam hal pengalokasian sumber daya yang sering dijumpai dalam kehidupan sehari-hari seperti dalam kasus pemilihan media promosi. Universitas Multimedia Nusantara (UMN) dalam memilih media promosi perlu mengoptimasikan sumber daya yang ada untuk memilih media promosi dengan perolehan audiens yang paling banyak.**

Penelitian ini menghasilkan sebuah aplikasi berbasis Java untuk melakukan optimasi sumber daya dalam pemilihan media promosi bagi UMN. Hasil uji coba menyatakan bahwa penggunaan algoritma *Dynamic Programming* untuk mendapatkan solusi optimal atas permasalahan MCKP dapat diimplementasikan pada optimasi pemilihan media promosi. Aplikasi ini mempermudah pengguna dalam memilih media promosi yang akan digunakan, karena media promosi yang optimal dapat ditentukan dalam waktu yang lebih singkat sekaligus lebih akurat dibandingkan dengan optimasi secara manual.

Kata kunci—Knapsack Problem; Multiple Constraints Knapsack Problem (MCKP); Dynamic Programming; optimasi; pemilihan media promosi

I. PENDAHULUAN

Masalah *knapsack* adalah permasalahan optimasi yang mendasar. Masalah ini merupakan permasalahan algoritma yang sudah dikenal dengan luas. Dalam kehidupan sehari-hari masalah ini seringkali dijumpai ketika hendak memilih sebuah solusi atas suatu permasalahan optimasi kombinatorik. Permasalahan ini bertambah kompleks, ketika tiap-tiap pilihan yang ada masing-masing memiliki lebih dari satu dimensi batasan yang lebih dikenal dengan nama "*Multiple Constraints Knapsack Problem*" [9].

Pada bidang pemasaran tersedia macam-macam pilihan media promosi yang dapat digunakan untuk menawarkan produk atau jasa kepada masyarakat. Dalam memilih media promosi terdapat lebih dari satu dimensi batasan yang dijadikan sebagai bahan pertimbangan. Kendala yang berhubungan dengan batasan-batasan tersebut adalah adanya keterbatasan pada sumber daya yang dimiliki. Sehingga perlu dilakukan optimalisasi penggunaan sumber daya yang tersedia untuk meraih perolehan konsumen yang paling banyak. Untuk melakukan hal ini diperlukan kalkulasi manual yang cermat agar menghasilkan pilihan yang maksimal.

Pendekatan untuk pemecahan masalah knapsack yang cukup terkenal adalah "*greedy approach*", yaitu untuk setiap kemungkinannya dicoba satu per satu, akan tetapi membutuhkan tenaga dan waktu yang besar. Sehingga pendekatan ini jarang digunakan terutama dengan adanya algoritma *Dynamic Programming*. Dengan algoritma ini, masalah knapsack dapat dipecahkan secara jauh lebih cepat dibanding dengan menggunakan *greedy approach* [2].

Perkembangan teknologi dewasa ini memungkinkan pencarian solusi optimal tersebut untuk dilakukan oleh aplikasi komputer. Dengan bantuan komputer, proses optimasi juga menjadi lebih cepat dan tepat dibandingkan dikerjakan secara manual oleh manusia. Sehingga penggunaan komputer sangat cocok untuk diimplementasikan pada bidang pemasaran khususnya untuk memilih media promosi dan pada bidang-bidang dengan permasalahan serupa lainnya.

Universitas Multimedia Nusantara (UMN) merupakan salah satu universitas berbasiskan teknologi multimedia yang terus bertumbuh dan berkembang dengan pesat dewasa ini. Hal tersebut tentunya perlu didukung dengan penggunaan media promosi yang tepat. Dalam pemilihan media promosi, UMN memiliki pertimbangan secara kuantitatif seperti contohnya biaya dan secara kualitatif seperti contohnya citra untuk masing-masing tipe media promosi.

Pertimbangan secara kuantitatif tersebut harus dapat memenuhi ketersediaan sumber daya yang ada. Pengalokasian sumber daya yang tidak optimal tentunya akan sangat merugikan, karena berarti berkurangnya kesempatan untuk menjangkau jumlah calon mahasiswa yang maksimal.

Pertimbangan secara kuantitatif dalam pemilihan media promosi tidak terbatas hanya berdasarkan batasan biaya saja. Tetapi juga harus mempertimbangkan batasan-batasan lainnya seperti jumlah konsumen untuk masing-masing media, banyaknya jumlah tenaga atau pegawai yang dibutuhkan untuk mengurus promosi di masing-masing media, serta waktu

yang dibutuhkan untuk membuat materi promosi sampai dimuat di media yang dimaksud. Dengan banyaknya batasan yang harus dipertimbangan maka proses optimasi sumber daya pemasaran menjadi lebih rumit. Belum lagi jika ditambah dengan pertimbangan-pertimbangan yang bersifat kualitatif lainnya. Hal ini menambah tingkat kerumitan untuk memilih media promosi yang optimal.

Penelitian ini dilakukan untuk menghasilkan aplikasi optimasi untuk mencari solusi optimal dari MCKP 0-1 dengan studi kasus pemilihan media promosi di UMN. Selain itu juga untuk mengimplementasikan algoritma *Dynamic Programming*.

Permasalahan yang diteliti dan diuraikan dalam penelitian ini adalah bagaimana melakukan pencarian solusi optimal pada *Multiple Constraints Knapsack Problem* (MCKP) dengan algoritma *Dynamic Programming* untuk aplikasi pemilihan media promosi di UMN. Selain itu adalah bagaimana mengimplementasikan pencarian solusi optimal dengan menggunakan algoritma *Dynamic Programming* pada aplikasi pemilihan media promosi di UMN.

Metode yang digunakan dalam penelitian ini antara lain adalah studi literatur, wawancara, perancangan sistem, implementasi, dan pengujian. Studi Literatur yaitu melakukan studi kepustakaan terhadap berbagai referensi yang berkaitan dengan penelitian yang dilakukan. Topik-topik yang dikaji antara lain meliputi: algoritma *Dynamic Programming*, *knapsack problem*, *multiple constraints knapsack problem,* pemasaran, dan media promosi. Wawancara dilaksanakan terhadap dengan divisi marketing di UMN. Topik-topik yang dikaji antara lain meliputi: pemilihan media promosi di UMN, sumber daya *marketing* di UMN, pertimbangan kuantitatif yang digunakan oleh UMN, dan media promosi yang selama ini digunakan oleh UMN. Perancangan Sistem yaitu melakukan analisis terhadap masalah yang dihadapi dan merancang sistem untuk mengimplementasikan algoritma *Dynamic Programming*. Kemudian melakukan implementasi algoritma *Dynamic Programming* pada sistem. Terakhir melakukan pengujian terhadap aplikasi yang telah dibuat. Tujuannya adalah untuk memperbaiki *defect* yang ditemukan pada aplikasi. *Defect* disini merupakan *block of code* yang tidak sesuai dengan rancangan ataupun yang tidak berfungsi sebagaimana harusnya.

II. TINJAUAN PUSTAKA

A. *Knapsack Problem*

Masalah knapsack adalah permasalahan optimasi kombinatorial dimana satu set *item*, masing-masing memiliki beban dan nilai, menentukan jumlah dari setiap *item* yang akan disertakan dalam koleksi sehingga beban total kurang dari atau sama dengan beban dan dengan nilai total sebesar mungkin. Secara matematik masalah *knapsack* dapat diformulasikan sebagai Formulasi *Knapsack Problem* 0-1 berikut ini.

$$\text{maximize} \sum_{i=1}^{n} v_i x_i \quad (1)$$

$$\text{subject to} \sum_{i=1}^{n} w_i x_i \leqslant W, \qquad x_i \in \{0,1\} \quad (2)$$

Formulasi yang paling sering dan umum dari masalah *knapsack* adalah *Knapsack Problem* 0-1, dimana membatasi kemungkinan pilihan yang diambil antara 1 atau 0 [4].

Adapun versi lain dari tipe *knapsack* adalah *fractional knapsack problem,* dimana *item* yang dipilih dapat diambil tidak seluruhnya, tetapi dalam pecahan-pecahan tertentu [4]. Versi *knapsack* 0-1 sebelumnya, dapat diselesaikan dengan algoritma *Dynamic Programming*. Sedangkan untuk versi *fractional knapsack problem*, dapat diselesaikan menggunakan *greedy algorithm* [6].

The Bounded Knapsack Problem membatasi jumlah duplikasi dari tiap pilihan menjadi nilai integer maksimum. Secara matematik *Bounded Knapsack Problem* dapat diformulasikan sebagai Formulasi *Bounded Knapsack Problem* berikut ini.

$$\text{maximize} \sum_{i=1}^{n} v_i x_i \quad (1)$$

$$\text{subject to} \sum_{i=1}^{n} w_i x_i \leqslant W, \qquad x_i \in \{0,1,\ldots,c_i\} \quad (2)$$

The Unbounded Knapsack Problem (UKP) tidak memberikan batasan terhadap jumlah duplikasi dari tiap pilihan. Perumusannya dapat menggunakan rumus diatas juga, tetapi untuk X_i harus bilangan bulat positif.

1) Multiple Constraints Knapsack Problem

*Multiple Constraints Knapsack Problem (*MCKP*)* adalah suatu permasalahan *knapsack* yang sering disebut juga dengan nama "*Multidimensional Knapsack Problem*" (MKP). Dimana MCKP ini sendiri merupakan permasalahan optimasi kombinatorial NP-*hard* yang terdapat pada beragam aplikasi [10].

Pada MCKP, tiap-tiap *item* pilihan memiliki batasan lebih dari satu dimensi. Batasan-batasan dalam memilih suatu media promosi contohnya seperti biaya, waktu, dan pekerja. Tujuan yang ingin dicapai pada permasalahan ini adalah memperoleh solusi optimum dengan memilih kombinasi *item* sedemikian rupa dimana semua batasan juga tidak melewati kapasitas yang tersedia [8].

Secara matematik, *Multiple Constraints Knapsack Problem* dapat dirumuskan sebagai Formulasi MCKP berikut ini [10].

$$\text{maximize} \quad z = \sum_{j=1}^{n} p_j x_j \quad (1)$$

$$\text{subject to} \quad \sum_{j=1}^{n} w_{ij} x_j \leq c_i, \quad i = 1,\ldots,m, \quad (2)$$

$$x_j \in \{0,1\}, \quad j = 1,\ldots,n. \quad (3)$$

Parameter-parameter yang digunakan untuk perumusan MCKP diatas :

P_j : *Profit* tiap item j ($P_j > 0$)
X_j : *Decision variables* (1 jika dipilih dan 0 jika tidak)
j : *Item*
W_{ij} : Nilai tiap *resource* i ($W_{ij} \geq 0$)
C_i : Kapasitas untuk tiap *resource* ($C_i > 0$)
i : *Resource*

m : Kumpulan *resource*
n : Kumpulan *item*

Perumusan matematik *Multiple Constraints Knapsack Problem* di atas memformulasikan perolehan nilai optimal (z) dari sekumpulan *items* n dengan *profits* P_j dan m *resources* dengan kapasitas C_i. Tiap *item* j mengkonsumsi sejumlah *amount* W_{ij} dari tiap *resource* i. *Decision variable* 0-1 (X_j) mengindikasikan *item-item* yang dipilih. Sesuai dengan (1), tujuan yang ingin diraih adalah memilih subset dari *items* dengan total profit yang maksimal. *Item-item* yang dipilih harus tidak boleh melebihi kapasitas *resources,* seperti didefinisikan pada *knapsack constraints* (2) [10].

B. Dynamic Programming

Dynamic Programming atau pemrograman dinamis adalah teknik desain yang mirip dengan *divide and conquer*. Algoritma *divide and conquer* mempartisi masalah menjadi *subproblem* independen, memecahkan *subproblem* secara rekursif, dan kemudian menggabungkan solusi untuk memecahkan masalah asli. Pemrograman dinamis dapat diterapkan ketika *subproblem* tidak independen. Algoritma *Dynamic Programming* memecahkan setiap *subsubproblem* hanya sekali dan kemudian menyimpan jawabannya dalam sebuah tabel, sehingga menghindari pekerjaan komputasi ulang untuk memperoleh jawabannya setiap kali *subsubproblem* diketemukan [4].

Pemrograman dinamis biasanya diterapkan untuk masalah optimasi. Dalam masalah seperti itu bisa ada banyak kemungkinan solusi. Setiap solusi memiliki nilai, dan kita ingin mencari solusi dengan nilai optimal (minimum atau maksimum). Seringkali disebut sebagai solusi-solusi optimal, dibandingkan dengan solusi optimal, karena mungkin ada beberapa solusi yang mencapai nilai optimal [4].

Pembangunan algoritma *Dynamic Programming* dapat dipecah menjadi empat langkah yang berurutan yaitu pertama adalah karaterisasi struktur dari solusi optimal, kedua adalah secara rekursif mendefinisikan nilai dari solusi optimal, ketiga adalah menghitung nilai dari solusi optimal secara *bottom-up*, dan keempat adalah membuat sebuah solusi optimal berdasarkan informasi hasil komputasi.

Langkah 1-3 membentuk landasan solusi algoritma *Dynamic Programming* terhadap masalah. Langkah 4 dapat dilewati apabila hanya nilai dari solusi optimal yang ingin diperoleh [4].

Berikut ini adalah relasi rekurensi untuk permasalahan *Knapsack* [5].

If $j < w_i$ then
  Table[i, j] ← Table[i-1, j] — *Cannot fit* the i^{th} item
Else
  Table[i, j] ← *maximum* { Table[i-1, j] — *Do not use* the i^{th} item
  AND
  v_i + Table[i-1, j – v_i,] — *Use* the i^{th} item

Gambar 2.1 Relasi Rekurensi *Knapsack Problem*

Berikut ini adalah *pseudocode* algoritma *Dynamic Programming* untuk permasalahan *Knapsack* [5].

```
ALGORITHM Dynamic Programming (Weights [1 … N], Values [1 … N],
                               Table [0 ... N, 0 … Capacity])
// Input:  Array Weights contains the weights of all items
           Array Values contains the values of all items
           Array Table is initialized with 0s; it is used to store the results from the dynamic
           programming algorithm.
// Output: The last value of array Table (Table [N, Capacity]) contains the optimal
           solution of the problem for the given Capacity

for i = 0 to N do
    for j = 0 to Capacity
        if  j < Weights[i]      then
                Table[i, j] ← Table[i-1, j]
        else
                Table[i, j] ← maximum { Table[i-1, j]
                                        AND
                                Values[i] + Table[i-1, j – Weights[i]]
return Table[N, Capacity]
```

Gambar 2.2 *Pseudocode Dynamic Programming*

Berikut ini adalah pseudocode untuk mengetahui *item* yang termasuk dalam solusi [5].

```
n ← N          c ← Capacity
Start at position Table[n, c]
While the remaining capacity is greater than 0 do
        If Table[n, c] = Table[n-1, c] then
                Item n has not been included in the optimal solution
        Else
                Item n has been included in the optimal solution
                Process Item n
                Move one row up to n-1
                Move to column c – weight(n)
```

Gambar 2.3 *Pseudocode* Komposisi *Item* Solusi Optimal

C. Pemasaran

Pemasaran menurut Cannon, Perreault, dan McCarthy adalah suatu kegiatan yang berusaha untuk mencapai tujuan organisasi dengan mengantisipasi kebutuhan pelanggan atau klien dan mengarahkan aliran atas kebutuhan barang atau jasa yang memuaskan dari produsen ke pelanggan atau klien. Konsep pemasaran itu sendiri adalah ide dimana sebuah organisasi harus mengarahkan seluruh *effort*-nya untuk memuaskan pelanggan dan juga memperoleh keuntungan.

Advertising adalah segala bentuk pembayaran atas presentasi nonpersonal untuk ide, barang, atau pelayanan oleh sebuah sponsor yang dikenali. Penentuan medium *advertising* tidaklah sederhana, efektivitas dari medium bergantung pada tujuan promosi, target *market* yang ingin dicapai, biaya yang tersedia untuk *advertising*, dan sifat dasar dari media tersebut [3].

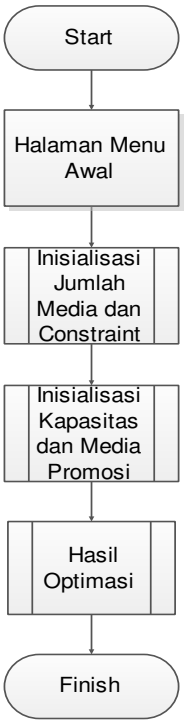
1) Media Promosi

Media promosi menurut Philip Kotler dan Gary Armstrong adalah sebuah kendaraan dimana pesan promosi disampaikan kepada audiens yang dituju. Media promosi terbagi dalam beberapa tipe besar yaitu televisi, koran, internet, *direct mail*, majalah, radio, dan *outdoor*. Secara umum yang menjadi permasalahan adalah bukan tipe media yang dipilih melainkan pengkombinasian media-media tersebut dan

mencampurkannya menjadi sebuah komunikasi pemasaran yang terintegrasi secara utuh.

III. PERANCANGAN SISTEM

Sistem dirancang dengan menggunakan *flowchart*. *Flowchart* yang dirancang meliputi *flowchart* Aplikasi, *flowchart Subroutine* Inisialisasi Jumlah Media dan Constraint, *flowchart Subroutine* Inisialisasi Kapasitas dan Media Promosi, dan *flowchart Subroutine* Hasil Optimasi.

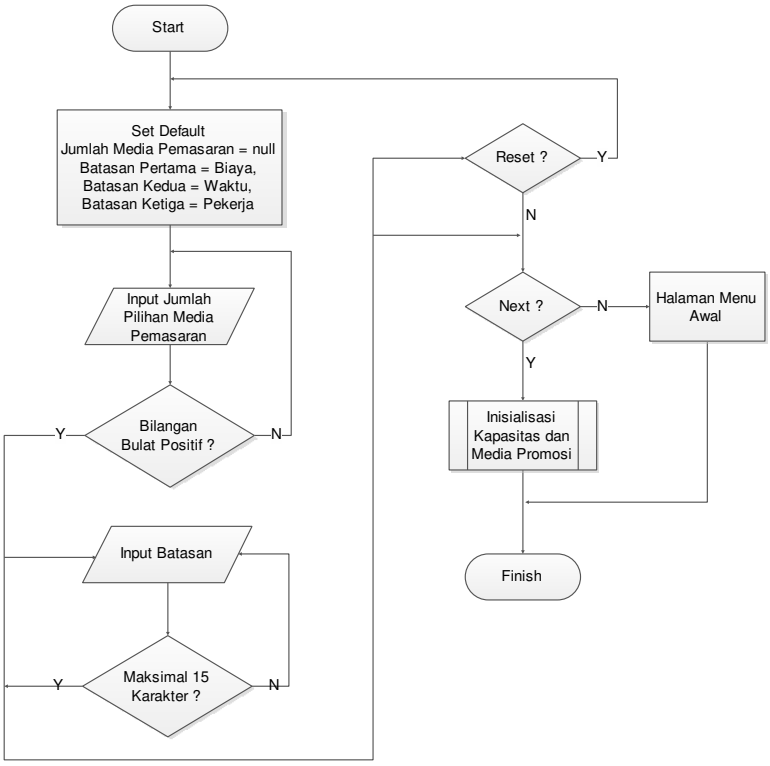


Gambar 3.1 *Flowchart* Aplikasi

Berikutnya yaitu *flowchart Subroutine* Inisialisasi Jumlah Media dan Constraint mewajibkan pengguna untuk memasukkan *input* berupa jumlah pilihan media promosi. *Subroutine* secara *default* telah mengatur nama untuk batasan pertama yaitu biaya, nama batasan kedua yaitu waktu, dan nama batasan ketiga yaitu pekerja. Adapun ketiga batasan tersebut dapat diubah namanya oleh pengguna.

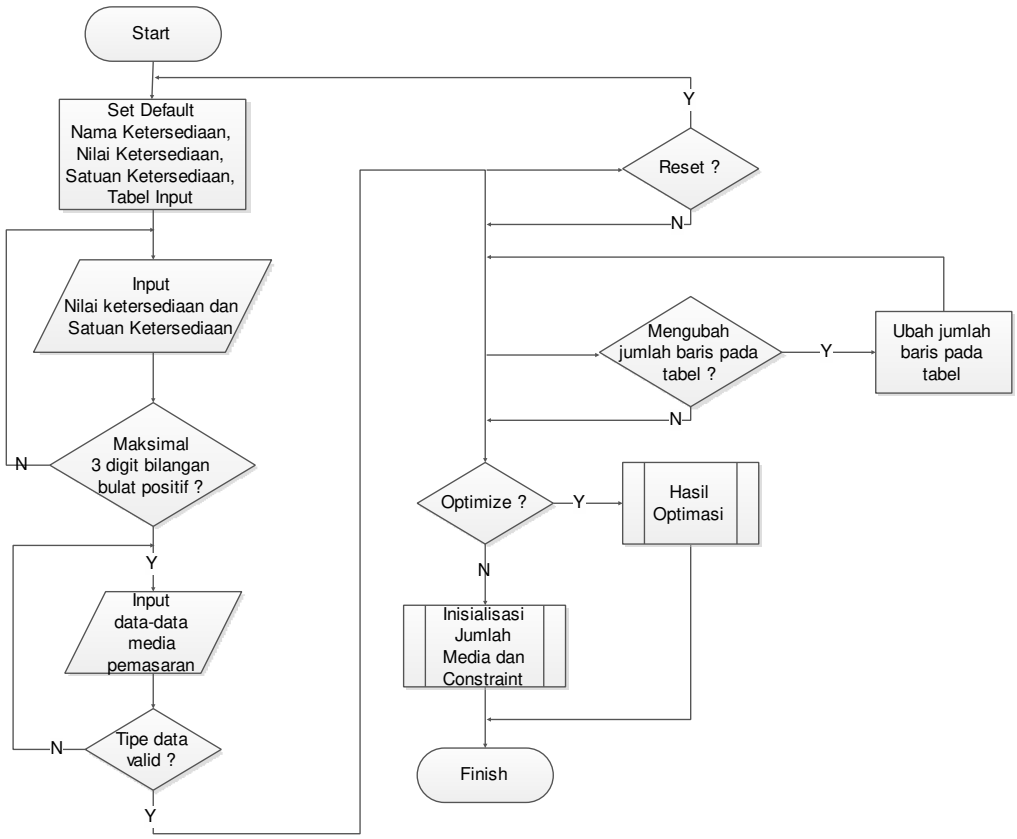
Flowchart Aplikasi menampilkan halaman Menu Awal dan kemudian meminta input dari pengguna berupa jumlah pilihan media promosi dan nama batasan. Selanjutnya aplikasi meminta input detail yaitu nilai-nilai dari tiap batasan untuk tiap media promosi, dan kapasitas dari tiap batasan beserta dengan satuan yang digunakan. Terakhir aplikasi menampilkan solusi optimal dan media promosi yang digunakan. Berikut adalah *flowchart* Aplikasi.

Subroutine kemudian melakukan validasi berupa pengecekan *input* untuk jumlah media promosi dan tiap-tiap batasan. Jumlah media promosi yang didefinisikan oleh pengguna harus berupa bilangan bulat positif dengan jumlah digit maksimal 3 buah. Sedangkan validasi nama tiap batasan adalah memiliki jumlah karakter maksimal 15 buah. Berikut adalah gambar dari *flowchart Subroutine* Inisialisasi Jumlah Media dan Constraint.



Gambar 3.2 *Flowchart Subroutine* Inisialisasi Jumlah Media dan *Constraint*

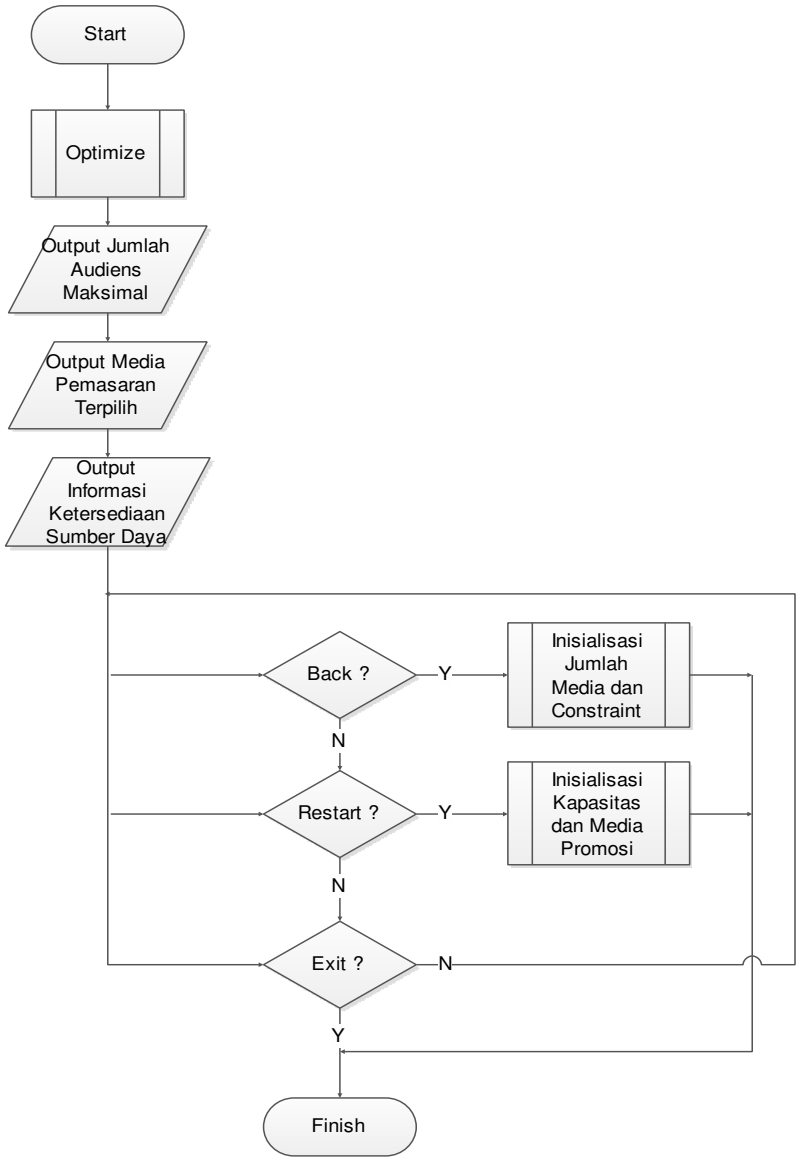
Berikutnya adalah *flowchart Subroutine* Inisialisasi Kapasitas dan Media Promosi. Dibawah ini merupakan gambarnya.



Gambar 3.3 *Flowchart Subroutine* Inisialisasi Kapasitas dan Media Promosi

Pada *subroutine* ini, pengguna diminta untuk memasukkan nilai ketersediaan untuk tiap-tiap batasan dan memilih satuan yang sesuai untuk tiap batasan tersebut. Pengguna juga diminta untuk menginputkan nama media promosi, nilai tiap-tiap batasan, dan jumlah audiens untuk tiap-tiap media promosi tersebut.

Flowchart Subroutine Hasil Optimasi melakukan komputasi dan menampilkan hasil komputasi tersebut kepada pengguna. Komputasi dilakukan di dalam *subroutine Optimize*. Berikut adalah gambar *flowchart Subroutine* Hasil Optimasi.

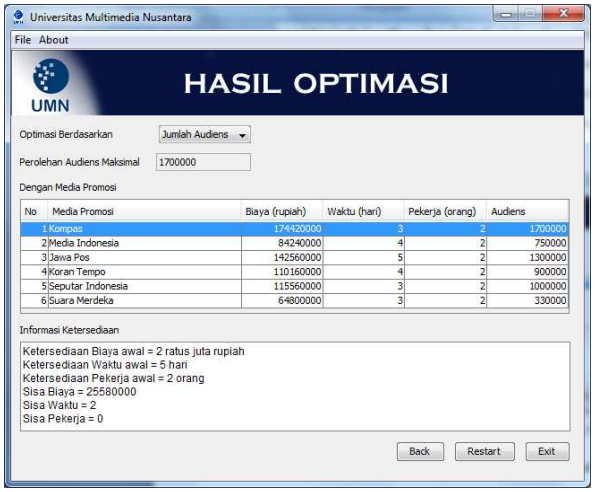


Gambar 3.4 *Flowchart Subroutine* Hasil Optimasi

IV. IMPLEMENTASI DAN PENGUJIAN

A. *Implementasi*

Aplikasi diuji coba menggunakan data-data yang dimasukkan secara langsung ke aplikasi. Data-data ini dimasukkan melalui *input field* yang telah disediakan pada halaman Form Awal dan halaman Form Detail. Implementasi dari pencarian solusi optimal pada aplikasi ini terdapat dalam halaman Hasil Optimasi. Dimana halaman ini dapat diakses setelah seluruh data yang dibutuhkan telah dimasukkan. Hasil optimasi yang ditampilkan pada halaman ini adalah nilai perolehan audiens optimal dan maksimal, serta media-media promosi yang termasuk dalam nilai optimal atau maksimal tersebut. Berikut adalah tampilan dari halaman Hasil Optimasi. Secara *default* optimasi yang dilakukan adalah berdasarkan jumlah audiens terbanyak.



Gambar 4.1 Tampilan Hasil Optimasi Jumlah Audiens

Pada gambar diatas optimasi yang dilakukan adalah untuk meraih perolehan audiens terbanyak. Nilai yang digunakan dalam komputasi optimasi adalah murni nilai audiens. Pengguna dapat mengubah optimasi berdasarkan seluruh batasan. Optimasi juga dapat dilakukan berdasarkan tiap batasan seperti batasan biaya, batasan waktu, dan batasan pekerja.

B. Pengujian

Pengujian dilakukan untuk mengetahui seberapa akurat solusi yang dihasilkan oleh aplikasi. Selain itu juga untuk mengukur kecepatan waktu proses optimasi dari aplikasi. Pengujian keakuratan aplikasi dilakukan dengan dua buah tipe media promosi yaitu media koran cetak dan media *online*.

1) Evaluasi Keakuratan Optimasi

Uji coba pertama dilakukan dengan data masukkan dari pengguna berupa jumlah pilihan media promosi = 5, dengan batasan default. Biaya yang tersedia sebanyak 200.000.000 rupiah, waktu yang tersedia sebanyak 5 hari, dan pekerja yang tersedia sebanyak 2 orang. Proses optimasi yang dilakukan adalah berdasarkan jumlah audiens terbanyak. Spesifikasi detail untuk tiap-tiap media promosi dapat dilihat pada tabel 4.1 berikut ini.

TABEL 4.1 DATA PENGUJIAN MEDIA KORAN

No	Media Promosi	Biaya (rupiah)	Waktu (hari)	Pekerja (orang)	Audiens
1	Kompas	174.420.000	3	2	1.700.000
2	Media Indonesia	84.240.000	4	2	750.000 (estimasi)
3	Jawa Pos	142.560.000	5	2	1.300.000
4	Koran Tempo	110.160.000	4	2	900.000
5	Seputar Indonesia	115.560.000	3	2	1.000.000
6	Suara Merdeka	64.800.000	3	2	330.000 (estimasi)

Solusi optimal yang dihasilkan untuk permasalahan optimasi diatas terdiri atas beberapa pertimbangan. Pertama adalah solusi optimal berdasarkan jumlah perolehan audiens terbanyak yang dapat diperoleh. Kedua adalah solusi optimal berdasarkan optimasi penggunaan seluruh batasan. Ketiga adalah optimasi berdasarkan batasan pertama yaitu biaya. Keempat adalah optimasi berdasarkan batasan kedua yaitu waktu. Kelima adalah optimasi berdasarkan batasan ketiga yaitu pekerja.

Hasil optimasi untuk kelima solusi optimal yang dapat diperoleh adalah sebagai berikut. Solusi optimal berdasarkan perolehan jumlah audiens terbanyak, berdasarkan seluruh batasan, batasan biaya, batasan waktu, dan batasan pekerja juga untuk masing-masingnya adalah sama yaitu Kompas.

Tiap-tiap hasil solusi optimal yang telah dihasilkan oleh aplikasi kemudian dibandingkan dengan hasil yang diperoleh dengan menggunakan penelurusan secara manual. Pada pengujian pertama ini aplikasi telah menghasilkan solusi optimal yang akurat.

Uji coba kedua dilakukan dengan data masukkan dari pengguna berupa jumlah pilihan media promosi = 6, dengan batasan *default*. Biaya yang tersedia adalah sebesar 3.000.000 rupiah, waktu yang tersedia sebanyak 5 hari, dan pekerja yang tersedia sebanyak 5 orang. Dengan spesifikasi detail untuk tiap-tiap media promosi seperti pada tabel 4.2 berikut ini.

TABEL 4.2 DATA PENGUJIAN MEDIA *ONLINE*

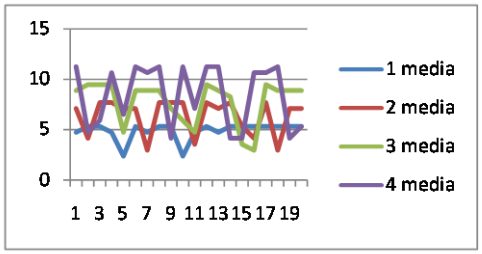
No	Media Promosi	Biaya (rupiah)	Waktu (hari)	Pekerja (orang)	Audiens
1	Google.com	750.000	1	2	2.991.250
2	Facebook.com	1.200.000	1	2	13.589.908
3	Yahoo.com	500.000	2	3	1.348.000
4	Kompas.com	1.500.000	2	2	1.209.156
5	Detik.com	1.500.000	2	3	2.799.810
6	Youtube.com	1.500.000	2	3	4.470.320

Dengan data pada tabel 4.2 dan menggunakan aplikasi maka diperoleh hasil-hasil optimal sebagai berikut. Solusi optimal berdasarkan perolehan jumlah audiens terbanyak, berdasarkan seluruh batasan, batasan biaya, batasan waktu, dan juga batasan pekerja adalah Google.com dan Facebook.com. Pada pengujian kedua ini aplikasi terbukti telah menghasilkan solusi optimal yang akurat.

2) Evaluasi Kecepatan Waktu Optimasi

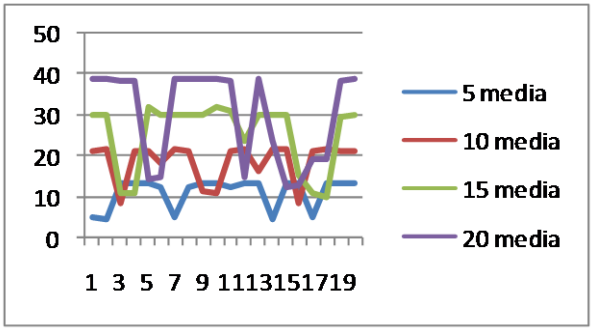
Evaluasi kecepatan waktu optimasi dilakukan untuk mengetahui kecepatan waktu dari proses optimasi yang dilakukan oleh aplikasi dan hubungannya dengan jumlah pilihan media promosi. Uji coba dilakukan sebanyak dua kali dengan menggunakan data yang serupa, satuan batasan biaya adalah juta rupiah, satuan batasan waktu adalah hari, dan satuan batasan pekerja adalah orang. Sampel hasil kecepatan waktu yang diambil adalah sebanyak 20 untuk tiap uji coba. Evaluasi ini bertujuan untuk mengetahui kecepatan waktu rata-rata, waktu tercepat, dan waktu terlama dari algoritma *Dynamic Programming* yang diimplementasikan.

Pengujian pertama dilakukan dengan menggunakan jumlah media promosi sebanyak 1 hingga 4 media. Hasil kecepatan waktu eksekusi algoritma untuk pengujian ini dapat dilihat pada gambar berikut.



Gambar 4.2 Grafik Perbandingan Waktu Pengujian Pertama

Sumbu Y pada grafik diatas menyatakan waktu dalam satuan *micro seconds* dan sumbu X pada grafik menyatakan sampel. Kemudian pengujian dilanjutkan dengan jumlah media promosi sebanyak 5, 10, 15, dan 20. Hasil kecepatan waktu eksekusi algoritma untuk pengujian ini dapat dilihat pada gambar berikut.



Gambar 4.3 Grafik Perbandingan Waktu Pengujian Kedua

Berdasarkan hasil pengujian pertama dan kedua diperoleh hasil sebagai berikut.

TABEL 4.3 HASIL PENGUJIAN KECEPATAN WAKTU ALGORITMA

Jumlah Media	Waktu Tercepat	Waktu Terlama	Waktu Rata-Rata
1	2,368	5,33	4,885
2	2,96	7,698	6,365
3	2,961	9,474	7,787
4	4,144	11,251	8,378
5	4,737	13,226	10,964
10	8,289	21,909	18,770
15	11,25	31,974	25,461
20	12,435	39,08	29,932

Pada hasil di tabel 4.3 terlihat dengan jelas bahwa waktu kecepatan algoritma berbanding lurus dengan banyaknya jumlah pilihan media promosi. Kecepatan algoritma juga sangat baik yaitu tercatat memiliki rata-rata waktu sebesar 29,932 *micro seconds* untuk optimasi dengan jumlah pilihan media sebanyak 20.

V. KESIMPULAN

Berdasarkan penelitian yang dilakukan maka dapat diperoleh kesimpulan sebagai berikut.

1. Telah dapat dihasilkan aplikasi optimasi untuk mencari solusi optimal dari MCKP 0-1 untuk studi kasus pemilihan media promosi di UMN.
2. Algoritma *Dynamic Programming* dapat diimplementasikan dalam aplikasi pemilihan media promosi.
3. Kecepatan waktu optimasi aplikasi berbanding lurus dengan jumlah pilihan media promosi.

DAFTAR PUSTAKA

[1] Bertsimas, D., & Demir, R, An Approximate Dynamic Programming Approach to Multidimensional Knapsack Problems, 2002.

[2] Biswajit, B, "Dynamic programming - its principles, applications, strenghts, and limitations," in International journal of implementing science and technology, 2010 , pp. 4822-4826.

[3] Cannon, J. P., Perreault, W. D., & McCarthy, E. J., Basic Marketing, Singapore: McGraw-Hill, 2008.

[4] Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C., Introduction to Algorithms. Cambridge: The MIT Press, 2007.

[5] Hristakeva, M., & Shrestha, D., Different Approaches to Solve the 0/1 Knapsack Problem.

[6] Jahangiri, E., & Ghassemi-Tari, F., "A dynamic programming approach for solving nonlinear knapsack problems" in Journal of industrial engineering international , 2006, pp. 31-37.

[7] Kotler, P., & Armstrong, G., Principles Of Marketing, Essex: Pearson Education, 2012.

[8] Lamine, A., Khemakhem, M., & Chabchoub, H., Knapsack Problems involving dimensions, demands and multiple choice constraints: generalization and transformations between formulations.

[9] Levitin, A., Introduction to The Design & Analysis of Algorithms, Boston: Pearson Addison Wesley, 2007.

[10] Puchinger, J., Raidl, G. R., & Pferschy, U., The Multidimensional Knapsack Problem: Structure and Algorithms, 2007.